Epistemic Strategies and Games on Concurrent Processes

Prakash Panangaden: Oxford University (on leave from McGill University).

Joint work with Sophia Knight, Konstantinos Chatzikokolakis and Catuscia Palamidessi.

Invited Talk at ICE 2011, Reykjavik.

My collaborators

How process algebra usually works

- Syntax for processes

How process algebra usually works

- Syntax for processes

$$P, Q ::== \mathbf{0} | a.P | P + Q | P \| Q | \nu x.P | !P$$

How process algebra usually works

- Syntax for processes

$$P, Q ::== \mathbf{0} | a.P | P + Q | P \| Q | \nu x.P | !P$$

- Operational Semantics

How process algebra usually works

- Syntax for processes

$$P, Q ::== \mathbf{0} | a.P | P + Q | P \| Q | \nu x.P | !P$$

- Operational Semantics

$$\frac{p_1 \xrightarrow{b_1} p_1' \quad \ldots \quad p_n \xrightarrow{b_n} p_n'}{p \xrightarrow{a} p'}$$

How process algebra usually works

- Syntax for processes

$$P, Q ::== \mathbf{0} | a.P | P + Q | P \| Q | \nu x.P | !P$$

- Operational Semantics

$$\frac{p_1 \xrightarrow{b_1} p_1' \quad \ldots \quad p_n \xrightarrow{b_n} p_n'}{p \xrightarrow{a} p'}$$

- Nondeterminism is resolved by an *omniscient*, *omnipotent* and *invisible* **scheduler**.

Reasoning about security using process algebra

Reasoning about security using process algebra

Model the system in your favourite process algebra

Reasoning about security using process algebra

Model the system in your favourite process algebra

Formulate the desired property, perhaps in some modal logic.

Reasoning about security using process algebra

Model the system in your favourite process algebra

Formulate the desired property, perhaps in some modal logic.

Show that the property holds under *all* choices of scheduler.

Reasoning about security using process algebra

Model the system in your favourite process algebra

Formulate the desired property, perhaps in some modal logic.

Show that the property holds under *all* choices of scheduler.

But if the scheduler is omniscient *and malicious*
it can leak information.

Reasoning about security using process algebra

Model the system in your favourite process algebra

Formulate the desired property, perhaps in some modal logic.

Show that the property holds under *all* choices of scheduler.

But if the scheduler is omniscient *and malicious*
it can leak information.

We need to describe the scheduler *explicitly*,

Reasoning about security using process algebra

Model the system in your favourite process algebra

Formulate the desired property, perhaps in some modal logic.

Show that the property holds under *all* choices of scheduler.

But if the scheduler is omniscient *and malicious*
it can leak information.

We need to describe the scheduler *explicitly*,

and perhaps to *restrict* its behaviour.

Example: Voting

Example: Voting

- Two candidates: a,b. Two voters: v,w.

Example: Voting

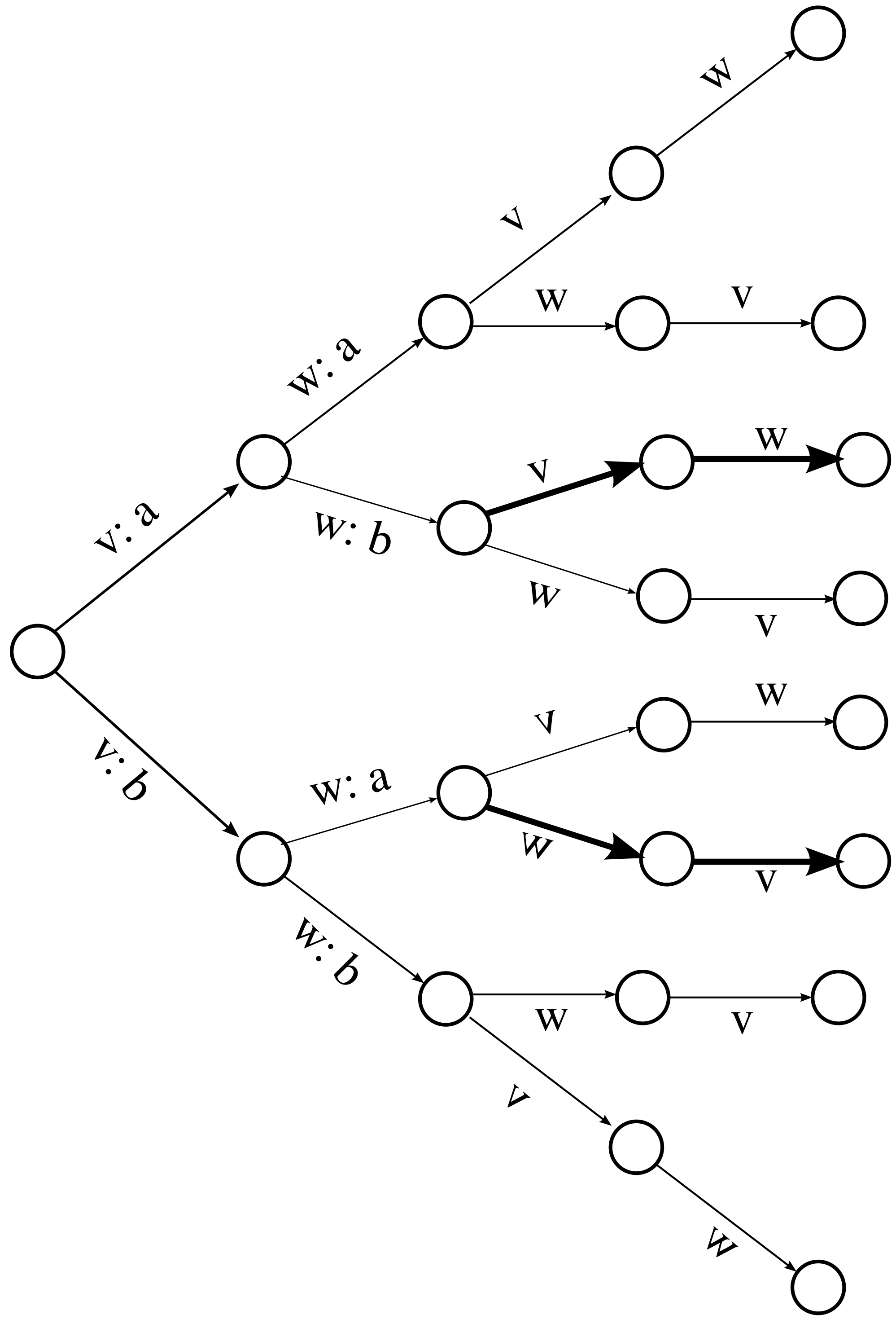
- Two candidates: a,b. Two voters: v,w.
- The system must reveal the list of people who actually voted (in any order) and the total votes for the candidate.

Example: Voting

- Two candidates: a,b. Two voters: v,w.
- The system must reveal the list of people who actually voted (in any order) and the total votes for the candidate.
- It must **not** reveal who voted for whom; unless the vote is unanimous.

Example: Voting

- Two candidates: a,b. Two voters: v,w.
- The system must reveal the list of people who actually voted (in any order) and the total votes for the candidate.
- It must **not** reveal who voted for whom; unless the vote is unanimous.
- A scheduler can leak the votes!



A scheduler that leaks voting preferences.

What do schedulers know?

What do schedulers know?

- The scheduler that resolves the nondeterminism in the order in which voters names are output should not "know" who voted for whom.

What do schedulers know?

- The scheduler that resolves the nondeterminism in the order in which voters names are output should not "know" who voted for whom.
- Chatzikokolakis and Palamidessi [CONCUR 07] described schedulers with an explicit syntax and operational semantics and used syntactic restrictions to control what scheduler knew.

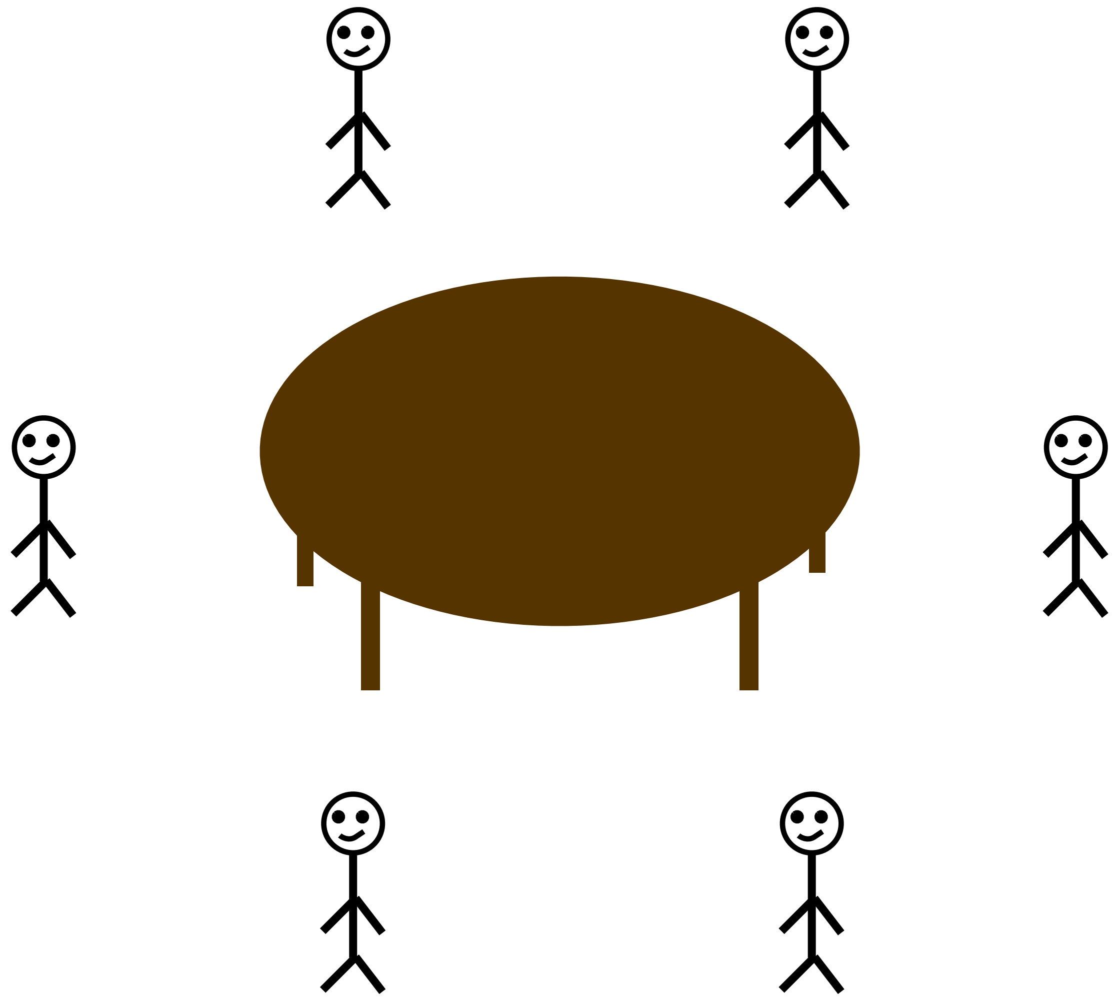
What do schedulers know?

- The scheduler that resolves the nondeterminism in the order in which voters names are output should not "know" who voted for whom.
- Chatzikokolakis and Palamidessi [CONCUR 07] described schedulers with an explicit syntax and operational semantics and used syntactic restrictions to control what scheduler knew.
- They had **two** schedulers to resolve different choices.

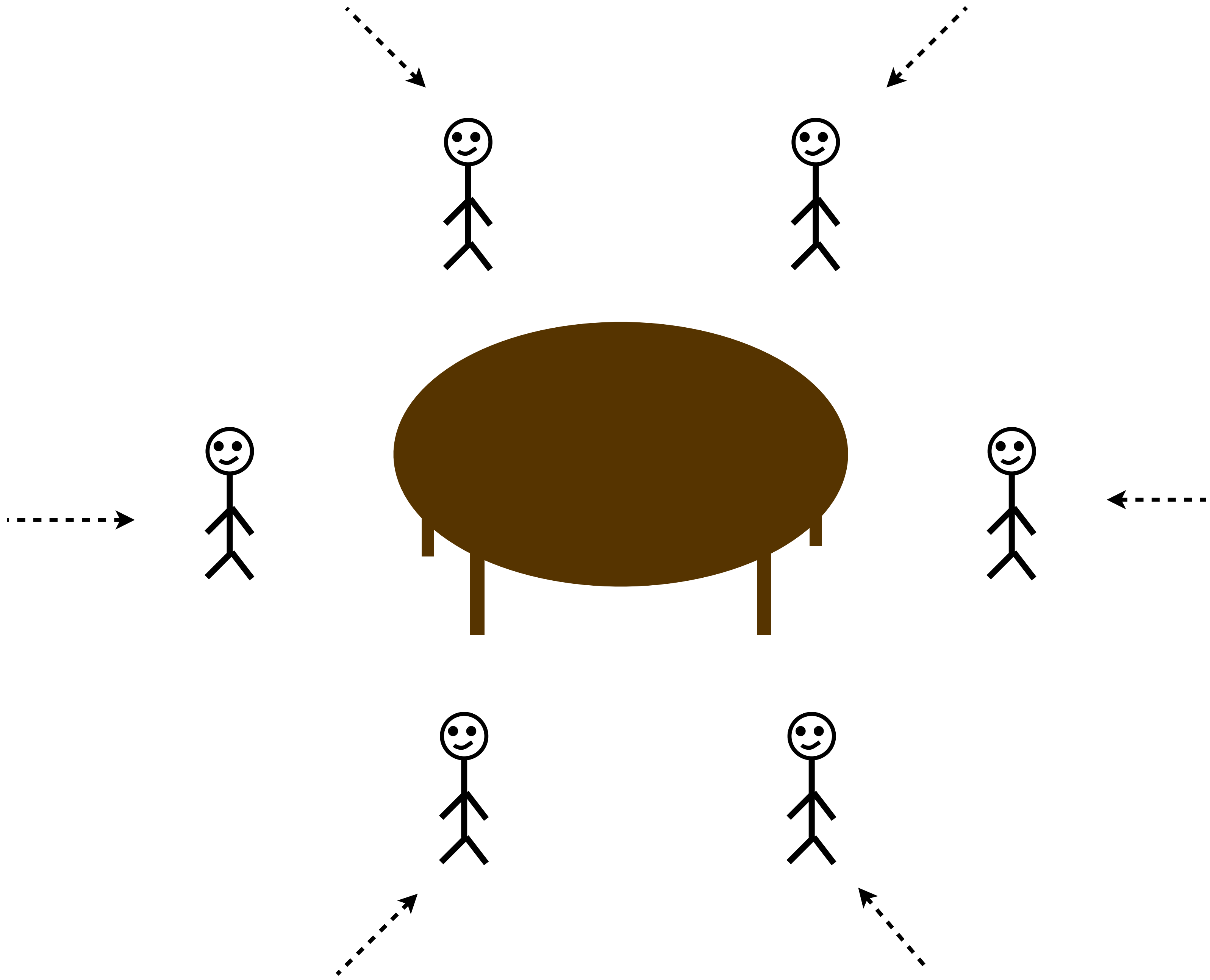
What do schedulers know?

- The scheduler that resolves the nondeterminism in the order in which voters names are output should not “know” who voted for whom.
- Chatzikokolakis and Palamidessi [CONCUR 07] described schedulers with an explicit syntax and operational semantics and used syntactic restrictions to control what scheduler knew.
- They had **two** schedulers to resolve different choices.
- They showed that certain equations hold.

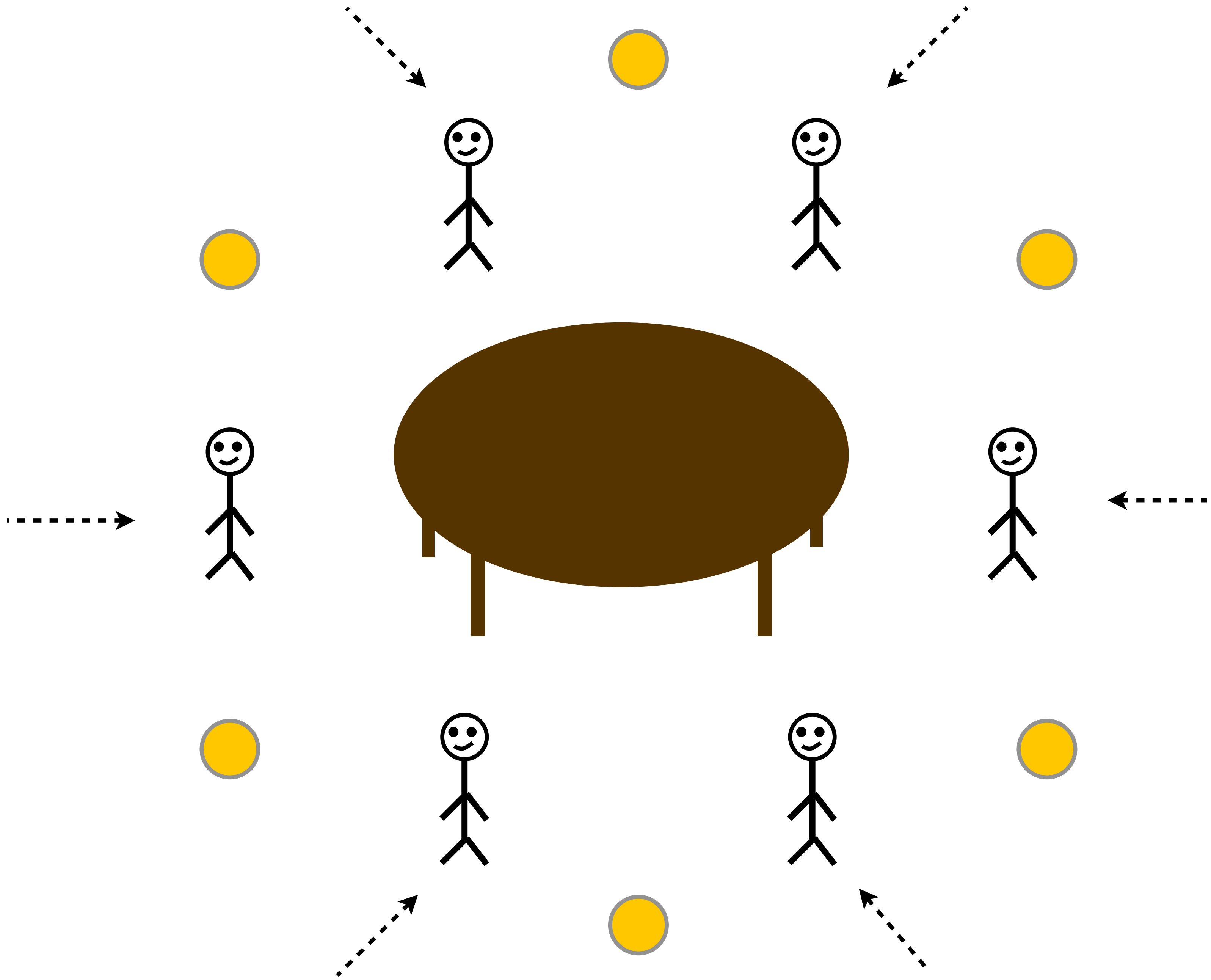
Dining Cryptographers



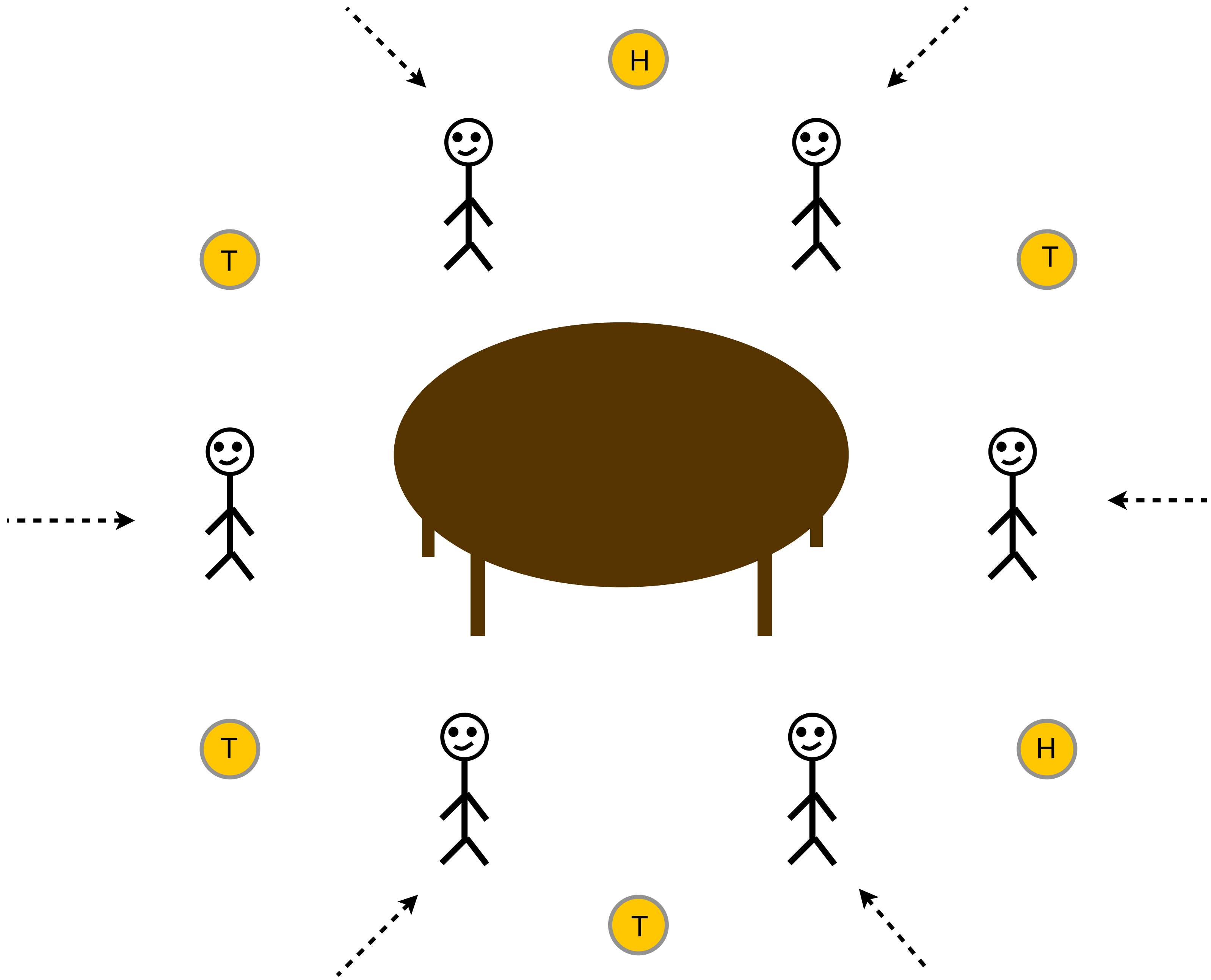
Dining Cryptographers



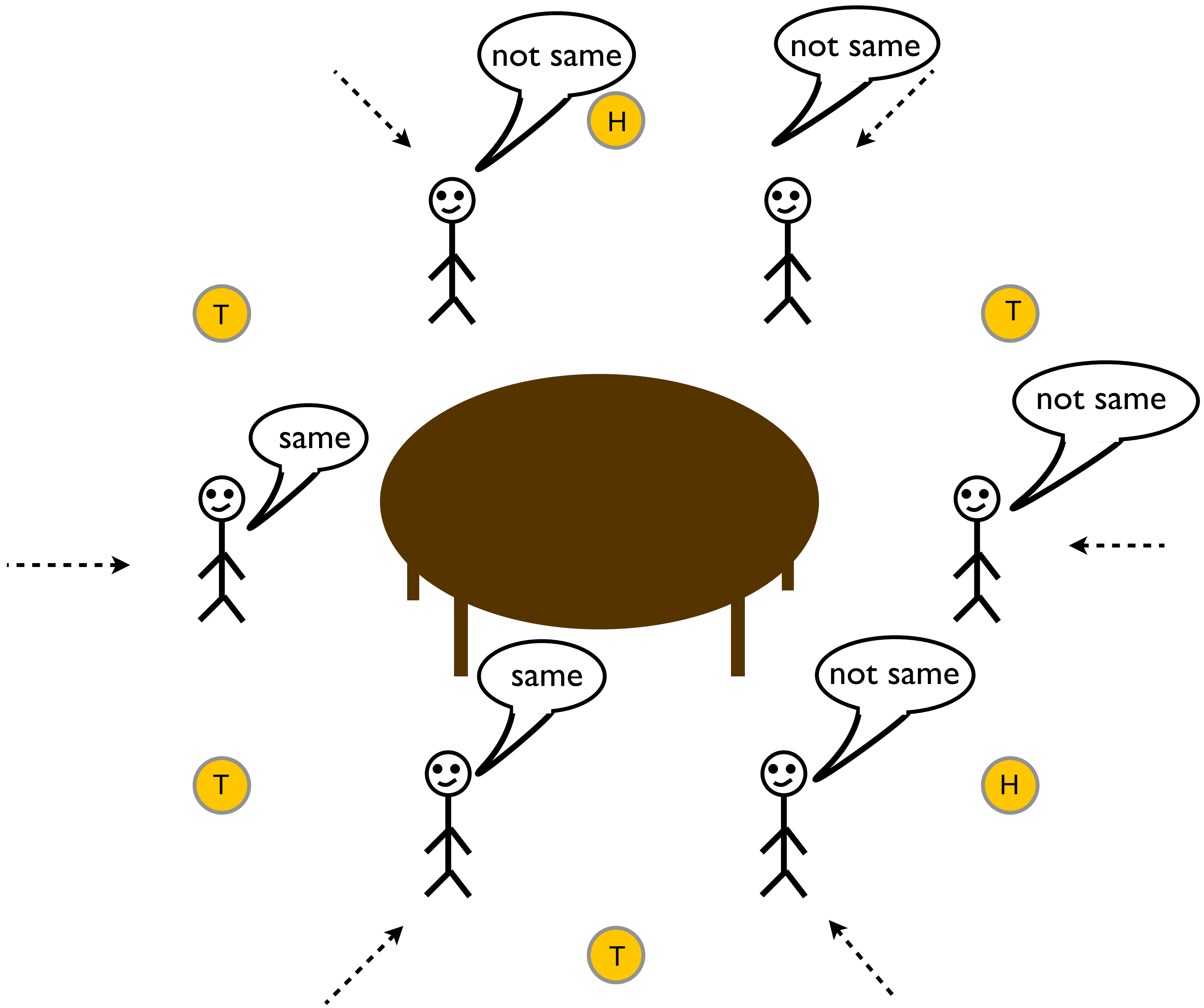
Dining Cryptographers



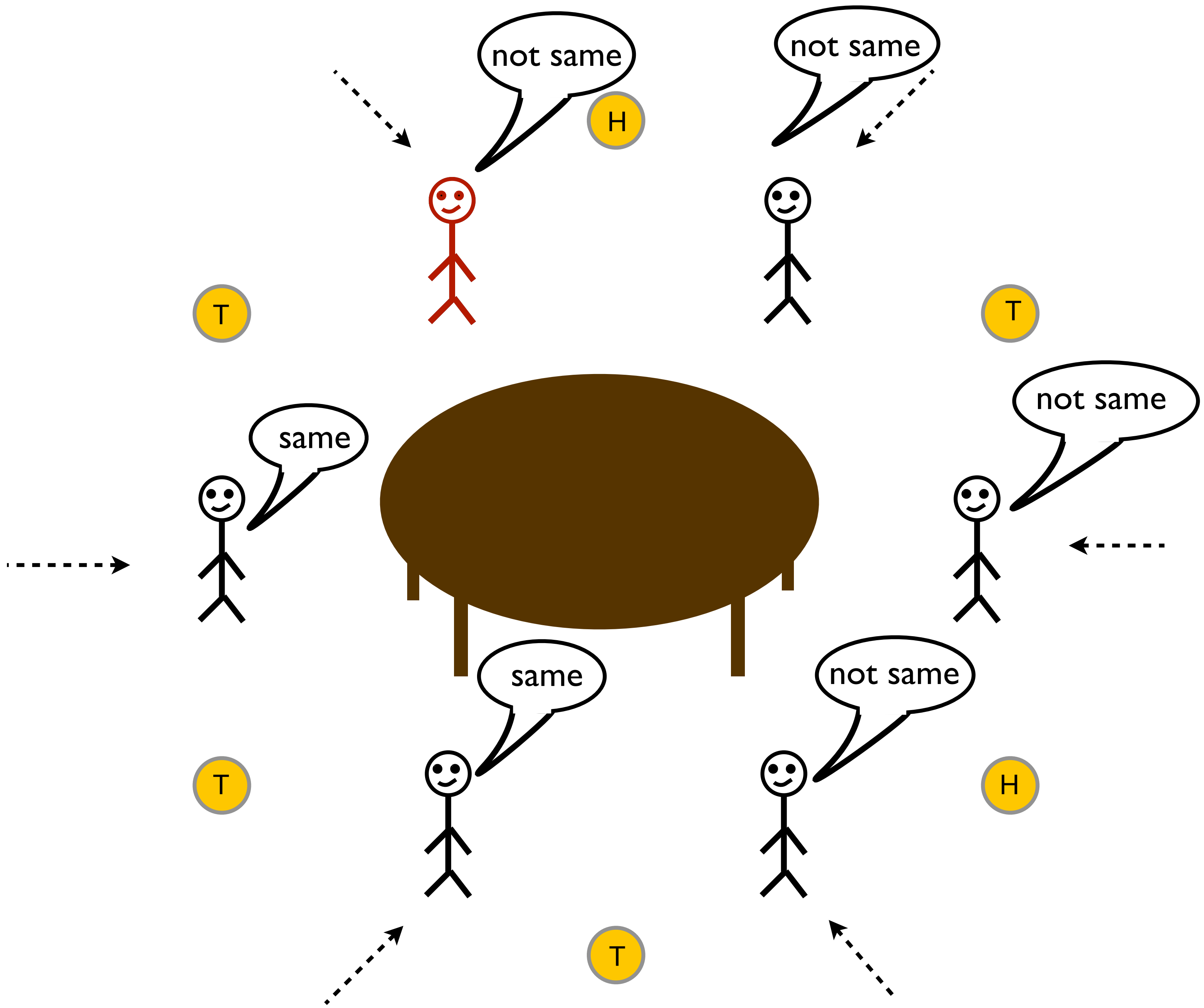
Dining Cryptographers



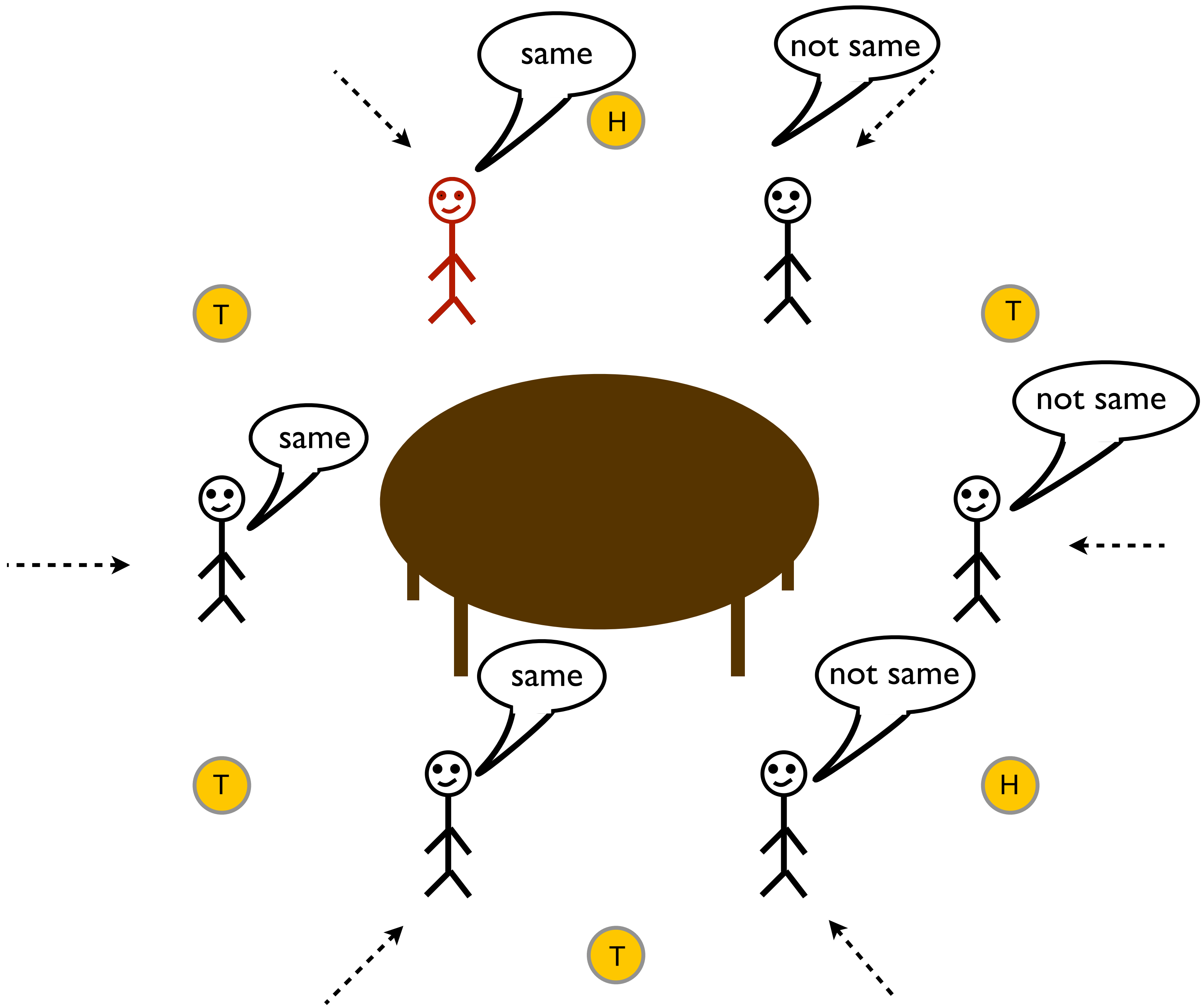
Dining Cryptographers



Dining Cryptographers



Dining Cryptographers



Why does it work?

Why does it work?

- View H = 0 and T = 1, same = 0 and not same = 1.

Why does it work?

- View H = 0 and T = 1, same = 0 and not same = 1.
- Then each person announces (left coin + right coin) mod 2 so if no one lies, the sum should be 0.

Why does it work?

- View H = 0 and T = 1, same = 0 and not same = 1.
- Then each person announces (left coin + right coin) mod 2 so if no one lies, the sum should be 0.
- So the sum is 1 iff someone lies; i.e. there are an odd number of "not same" announcements iff someone is lying.

Why does it work?

- View H = 0 and T = 1, same = 0 and not same = 1.
- Then each person announces (left coin + right coin) mod 2 so if no one lies, the sum should be 0.
- So the sum is 1 iff someone lies; i.e. there are an odd number of “not same” announcements iff someone is lying.

The role of the scheduler

The role of the scheduler

- If we model this with process algebra then the scheduler has to schedule the order in which the announcements are made.

The role of the scheduler

- If we model this with process algebra then the scheduler has to schedule the order in which the announcements are made.
- An omniscient scheduler can decide always to schedule the liar (if there is one) last (or first).

The role of the scheduler

- If we model this with process algebra then the scheduler has to schedule the order in which the announcements are made.
- An omniscient scheduler can decide always to schedule the liar (if there is one) last (or first).
- This is very unreasonable; the scheduler should not “know” the outcomes of the coin tosses when making the schedule.

The role of the scheduler

- If we model this with process algebra then the scheduler has to schedule the order in which the announcements are made.
- An omniscient scheduler can decide always to schedule the liar (if there is one) last (or first).
- This is very unreasonable; the scheduler should not "know" the outcomes of the coin tosses when making the schedule.
- There are two kinds of nondeterminism and there should be two schedulers.

An example due to Chatzikokolakis and Palamidessi

An example due to Chatzikokolakis and Palamidessi

$$A \triangleq a(x).([x=0]\overline{ok} \\ +_{0.5} \\ [x=1]\overline{ok})$$

$$B \triangleq a(x).[x=0]\overline{ok} \\ +_{0.5} \\ a(x).[x=1]\overline{ok}$$

An example due to Chatzikokolakis and Palamidessi

$$A \triangleq a(x).([x = 0]\overline{ok} \; +_{0.5} \; [x = 1]\overline{ok}) \qquad B \triangleq a(x).[x = 0]\overline{ok} \; +_{0.5} \; a(x).[x = 1]\overline{ok}$$

If the random choices in A and B are private, we expect that $A \approx B$.

An example due to Chatzikokolakis and Palamidessi

$$A \triangleq a(x).([x=0]\overline{ok} \; +_{0.5} \; [x=1]\overline{ok}) \qquad\qquad B \triangleq a(x).[x=0]\overline{ok} \; +_{0.5} \; a(x).[x=1]\overline{ok}$$

If the random choices in A and B are private, we expect that $A \approx B$.

If we use the context $\overline{a}0|\overline{a}1$, then A can produce ok with probability $\frac{1}{2}$ **no matter what the scheduler does**.

An example due to Chatzikokolakis and Palamidessi

$$A \stackrel{\Delta}{=} a(x).([x=0]\overline{ok} \\ +_{0.5} \\ [x=1]\overline{ok})$$

$$B \stackrel{\Delta}{=} a(x).[x=0]\overline{ok} \\ +_{0.5} \\ a(x).[x=1]\overline{ok}$$

If the random choices in A and B are private, we expect that $A \approx B$.

If we use the context $\overline{a}0|\overline{a}1$, then A can produce ok with probability $\frac{1}{2}$ **no matter what the scheduler does**.

But with B, a scheduler that**knows** the outcome of the random choice can select the synchronization and make the probability of ok be 1 or 0.

In general when $+_p$ represents a private choice we would like to have

$$C[P +_p Q] \approx C[\tau.P] +_p C[\tau.Q]$$

In general when $+_p$ represents a private choice we would like to have

$$C[P +_p Q] \approx C[\tau.P] +_p C[\tau.Q]$$

This is an algebraic way of capturing the limited knowledge of the scheduler but it is very indirect.

Games and Knowledge

- Games are an ideal setting to explore epistemic concepts.
- Economists have been particularly active in developing these ideas.

Many types of games

Many types of games

- Games for verification: Luca de Alfaro, Henzinger, Chatterjee, Abramsky, Ong, Murawski,...

Many types of games

- Games for verification: Luca de Alfaro, Henzinger, Chatterjee, Abramsky, Ong, Murawski,...
- Games in economics: see, e.g. Adam Brandenburger's review on epistemic games.

Many types of games

- Games for verification: Luca de Alfaro, Henzinger, Chatterjee, Abramsky, Ong, Murawski,...
- Games in economics: see, e.g. Adam Brandenburger's review on epistemic games.
- Game semantics: Abramsky, Jagadeesan, Malacaria, Hyland, Ong, Nickau, Laird, McCusker...

Many types of games

- Games for verification: Luca de Alfaro, Henzinger, Chatterjee, Abramsky, Ong, Murawski,...
- Games in economics: see, e.g. Adam Brandenburger's review on epistemic games.
- Game semantics: Abramsky, Jagadeesan, Malacaria, Hyland, Ong, Nickau, Laird, McCusker...
- Games in logic: model theory, EF, Lorenzen,...

Many types of games

- Games for verification: Luca de Alfaro, Henzinger, Chatterjee, Abramsky, Ong, Murawski,...
- Games in economics: see, e.g. Adam Brandenburger's review on epistemic games.
- Game semantics: Abramsky, Jagadeesan, Malacaria, Hyland, Ong, Nickau, Laird, McCusker...
- Games in logic: model theory, EF, Lorenzen,...
- Games in hardware synthesis: Ghica

Games between schedulers.

Games between schedulers.

- In order to make the epistemic aspects more explicit we can think of schedulers as playing games.

Games between schedulers.

- In order to make the epistemic aspects more explicit we can think of schedulers as playing games.
- The concurrent process is the "board" and the moves end up choosing the action.

Games between schedulers.

- In order to make the epistemic aspects more explicit we can think of schedulers as playing games.
- The concurrent process is the "board" and the moves end up choosing the action.
- We control what the schedulers "know" by putting restrictions on the allowed strategies.

Restricting Strategies

Restricting Strategies

- What can an agent "see" in formulating its strategy? This controls what it "knows."

Restricting Strategies

- What can an agent "see" in formulating its strategy? This controls what it "knows."
- One possible restriction: an agent knows what choices are available to it **and what choices were available to it in the past.**

Restricting Strategies

- What can an agent "see" in formulating its strategy? This controls what it "knows."
- One possible restriction: an agent knows what choices are available to it **and what choices were available to it in the past.**
- This corresponds exactly to the CP syntactic restrictions [C,Knight, P 08].

Restricting Strategies

- What can an agent "see" in formulating its strategy? This controls what it "knows."
- One possible restriction: an agent knows what choices are available to it **and what choices were available to it in the past.**
- This corresponds exactly to the CP syntactic restrictions [C,Knight, P 08].
- Easy to impose epistemic restrictions on strategies.

Games and Concurrency

Games and Concurrency

- New direction in concurrency: Process algebras as defining interacting agents.

Games and Concurrency

- New direction in concurrency: Process algebras as defining interacting agents.
- Games are already used in many ways in concurrency, semantics, logic and economics.

Games and Concurrency

- New direction in concurrency: Process algebras as defining interacting agents.
- Games are already used in many ways in concurrency, semantics, logic and economics.
- But we still do not have a systematic way of describing and reasoning about interacting agents algebraically.

Processes with labels

a, b	actions
$\bar{a}, \bar{b}$	co-actions
τ	silent action
α, β	generic actions, co-actions, or silent action

$$P, Q \ ::= \ 0 \mid l : \alpha.P \mid P|Q \mid P + Q \mid (\nu a)P \mid l : \{P\}$$

Operational Semantics

$$\text{ACT}\ \frac{}{l{:}\alpha.P \xrightarrow{\alpha} P}$$

$$\text{RES}\ \frac{P \xrightarrow{\alpha} P' \qquad \alpha \neq a, \bar{a}}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'}$$

$$\text{SUM1}\ \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$$

$$\text{SUM2}\ \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$$

$$\text{PAR1}\ \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\text{PAR2}\ \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\text{COM}\ \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\text{SWITCH}\ \frac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$$

The SWITCH rule

The SWITCH rule

$$\mathrm{SWITCH}\ \frac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$$

The SWITCH rule

$$\text{SWITCH}\ \frac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$$

Represents the choices made independently of other choices in the process

The SWITCH rule

$$\text{SWITCH}\ \frac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$$

Represents the choices made independently of other choices in the process

Required to do a silent action because otherwise the outcome of the protected choice would be visible to the scheduler.

DEFINITION 2.1. *P is* deterministically labelled *if the following conditions hold:*

(1) *It is impossible for P to make two different transitions with the same labels: for all strings s, if* $P \xrightarrow[s]{\alpha} P'$ *and* $P \xrightarrow[s]{\beta} P''$ *then* $\alpha = \beta$ *and* $P' = P''$.

(2) *If* $P \xrightarrow[l_X . j_Y]{\tau} P'$ *then there is no transition* $P \xrightarrow[l_X]{\alpha} P''$ *for any* α *or* P''.

(3) *If* $P \xrightarrow[s]{\alpha} P'$ *then* P' *is deterministically labelled.*

Definition 2.1. *P is* deterministically labelled *if the following conditions hold:*

(1) *It is impossible for P to make two different transitions with the same labels: for all strings s, if $P \xrightarrow[s]{\alpha} P'$ and $P \xrightarrow[s]{\beta} P''$ then $\alpha = \beta$ and $P' = P''$.*

(2) *If $P \xrightarrow[l_X . j_Y]{\tau} P'$ then there is no transition $P \xrightarrow[l_X]{\alpha} P''$ for any α or P''.*

(3) *If $P \xrightarrow[s]{\alpha} P'$ then P' is deterministically labelled.*

$l : a.0 + l : b.0$ not deterministically labelled

Definition 2.1. *P is* deterministically labelled *if the following conditions hold:*

(1) *It is impossible for P to make two different transitions with the same labels: for all strings s, if $P \xrightarrow[s]{\alpha} P'$ and $P \xrightarrow[s]{\beta} P''$ then $\alpha = \beta$ and $P' = P''$.*

(2) *If $P \xrightarrow[l_X \cdot j_Y]{\tau} P'$ then there is no transition $P \xrightarrow[l_X]{\alpha} P''$ for any α or P''.*

(3) *If $P \xrightarrow[s]{\alpha} P'$ then P' is deterministically labelled.*

$l : a.0 + l : b.0$ not deterministically labelled

$l_1 : a.0 + l_1 : \{l_2 : \tau.0\}$ is not deterministically labelled

Definition 2.1. *P is* deterministically labelled *if the following conditions hold:*

(1) *It is impossible for P to make two different transitions with the same labels: for all strings s, if $P \xrightarrow[s]{\alpha} P'$ and $P \xrightarrow[s]{\beta} P''$ then $\alpha = \beta$ and $P' = P''$.*

(2) *If $P \xrightarrow[l_X . j_Y]{\tau} P'$ then there is no transition $P \xrightarrow[l_X]{\alpha} P''$ for any α or P''.*

(3) *If $P \xrightarrow[s]{\alpha} P'$ then P' is deterministically labelled.*

$l : a.0 + l : b.0$ not deterministically labelled

$l_1 : a.0 + l_1 : \{l_2 : \tau.0\}$ is not deterministically labelled

Also, $l_1 : a \mid l_2 : b \,.\, l_1 : c$ is not deterministically labelled because it can transition to $l_1 : a \mid l_1 : c$ which is not deterministically labelled.

Games

Games

- A game is defined for each specific process: the process is the game board.

Games

- A game is defined for each specific process: the process is the game board.
- Two-player games

Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction

Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction
  - the players are called X and Y

Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction
  - the players are called X and Y
- Players are independent and act according to their strategies.

Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction
  - the players are called X and Y
- Players are independent and act according to their strategies.
- Players interact to determine how process will execute.

Valid Positions

Valid Positions

- Players' moves are labels in the process.

Valid Positions

- Players' moves are labels in the process.
- A string of allowable moves is called a valid position.

Valid Positions

- Players' moves are labels in the process.
- A string of allowable moves is called a valid position.
- A valid position is like a trace, but with labels instead of actions.

Definition 3.1. *A* move *is anything of the form* l_X, l_Y, $(l,j)_X$, *or* $(l,j)_Y$ *where* l, *and* j *are labels.* l_X *and* $(l,j)_X$ *are called* X-moves *and* l_Y *and* $(l,j)_Y$ *are called* Y-moves

Definition 3.1. *A* move *is anything of the form* l_X, l_Y, $(l,j)_X$, *or* $(l,j)_Y$ *where* l, *and* j *are labels.* l_X *and* $(l,j)_X$ *are called* X-moves *and* l_Y *and* $(l,j)_Y$ *are called* Y-moves

Definition 3.2. *This extends the transition relation to multiple transitions, ignoring the actions for the transitions but keeping track of the labels.*

(1) *For any process* P, $P \xrightarrow[\varepsilon]{} P$.

(2) *If* $P \xrightarrow[s]{\alpha} P'$ *and* $P' \xrightarrow[s']{} P''$ *then* $P \xrightarrow[s.s']{} P''$.

Now we define valid positions.

Definition 3.3. *If* $P \xrightarrow[s]{} P'$ *then every prefix of* s *(including* s*) is a* valid position *for* P.

$$P = (\nu b)\big(l_1\!:\!\{k_1\!:\!\tau\,.\,l_2\!:\!a\,.\,l_3\!:\!b + k_2\!:\!\tau\,.\,l_2\!:\!c\,.\,l_3\!:\!b\} \mid l_4\!:\!\bar{b}\,.\,(l_5\!:\!d + l_6\!:\!e)\big).$$

Here are some of the valid positions for P:

$$l_{1X}.k_{1Y}.l_{2X}.(l_3, l_4)_X.l_{5X}$$
$$l_{1X}.k_{1Y}.l_{2X}.(l_3, l_4)_X.l_{6X}$$
$$l_{1X}.k_{2Y}.l_{2X}.(l_3, l_4)_X.l_{5X}$$
$$l_{1X}.k_{2Y}.l_{2X}.(l_3, l_4)_X.l_{6X}$$

Strategies

Strategies

Definition: In the game for P, a *strategy for player Z* is a set S of valid positions such that $\varepsilon \in S$ and if $s.m \in S$, then m is a Z move and every prefix of s ending with a Z move is in S.

Strategies

Definition: In the game for P, a *strategy for player* Z is a set S of valid positions such that $\varepsilon \in S$ and if $s.m \in S$, then m is a Z move and every prefix of s ending with a Z move is in S.

A strategy tells the player what move to make in a possible partial execution of the process.

$$P = (\nu b)\left(l_1\!:\!\{k_1\!:\!\tau\,.\,l_2\!:\!a\,.\,l_3\!:\!b + k_2\!:\!\tau\,.\,l_2\!:\!c\,.\,l_3\!:\!b\} \mid l_4\!:\!\bar{b}\,.\,(l_5\!:\!d + l_6\!:\!e)\right),$$

one strategy for X is:

$$\begin{array}{l}
\varepsilon \\
{l_1}_X \\
{l_1}_X.{k_2}_Y.{l_2}_X \\
{l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X \\
{l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X.{l_6}_X
\end{array}$$

Another strategy for X is:

$$\begin{array}{l}
\varepsilon \\
{l_1}_X \\
{l_1}_X.{k_1}_Y.{l_2}_X \\
{l_1}_X.{k_2}_Y.{l_2}_X
\end{array}$$

$$P = (\nu b)\left(l_1 : \{k_1 : \tau \,.\, l_2 : a \,.\, l_3 : b + k_2 : \tau \,.\, l_2 : c \,.\, l_3 : b\} \mid l_4 : \bar{b} \,.\, (l_5 : d + l_6 : e)\right),$$

one strategy for X is:

$$\begin{array}{l} \varepsilon \\ {l_1}_X \\ {l_1}_X . {k_2}_Y . {l_2}_X \\ {l_1}_X . {k_2}_Y . {l_2}_X . (l_3, l_4)_X \\ {l_1}_X . {k_2}_Y . {l_2}_X . (l_3, l_4)_X . {l_6}_X \end{array}$$

Another strategy for X is:

$$\begin{array}{l} \varepsilon \\ {l_1}_X \\ {l_1}_X . {k_1}_Y . {l_2}_X \\ {l_1}_X . {k_2}_Y . {l_2}_X \end{array}$$

Strategies need not be determinate.

$One\ strategy\ for\ Y\ \ is:$

$$\varepsilon$$
$$l_{1X}.k_{1Y}$$
$$l_{1X}.k_{2Y}$$

One strategy for Y is:

$$\begin{array}{l} \varepsilon \\ {l_1}_X.{k_1}_Y \\ {l_1}_X.{k_2}_Y \end{array}$$

This is not a strategy:

$$\begin{array}{l} \varepsilon \\ {l_1}_X \\ {l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X \end{array}$$

Restrictions 1

Definition A strategy S is *deterministic* if for all sequences s, $s.m_1 \in S$ and $s.m_2 \in S$ implies $m_1 = m_2$.

Complete strategies

- We want some way of ensuring that the strategy tells the player what to do in every possible situation.
- This is formalized by the definition of complete strategy.

DEFINITION 3.9. *Let V denote the set of valid positions for a process P. If s is a valid position for P, enabled(s) represents the set of moves available after s: define enabled$(s) = \{m | s.m \in V\}$. Also, define the X and Y moves available after s as, respectively, enabled$_X(s) = \{m_X | s.m_X \in V\}$ and enabled$_Y(s) = \{m_Y | s.m_Y \in V\}$.*

DEFINITION 3.12. *For a nonblocked process with valid positions V, a strategy S for player Z is* complete *if for all $s \in S$, for every string s' such that $Z(s') = \varepsilon$ and $s.s' \in V$ and enabled$_Z(s.s') \neq \emptyset$, then $s.s'.m \in S$ for some move m.*

Definition 3.9. *Let V denote the set of valid positions for a process P. If s is a valid position for P, enabled(s) represents the set of moves available after s: define enabled$(s) = \{m | s.m \in V\}$. Also, define the X and Y moves available after s as, respectively, enabled$_X(s) = \{m_X | s.m_X \in V\}$ and enabled$_Y(s) = \{m_Y | s.m_Y \in V\}$.*

Note that a position can have X moves enabled or Y moves enabled, but not both. This is clear from the operational semantics.

Definition 3.12. *For a nonblocked process with valid positions V, a strategy S for player Z is* complete *if for all $s \in S$, for every string s' such that $Z(s') = \varepsilon$ and $s.s' \in V$ and enabled$_Z(s.s') \neq \emptyset$, then $s.s'.m \in S$ for some move m.*

$$P = (\nu b)\left(l_1 : \{k_1 : \tau \,.\, l_2 : a \,.\, l_3 : b + k_2 : \tau \,.\, l_2 : c \,.\, l_3 : b\} \mid l_4 : \bar{b} \,.\, (l_5 : d + l_6 : e)\right),$$

$$\begin{array}{l} \varepsilon \\ {l_1}_X \\ {l_1}_X . {k_2}_Y . {l_2}_X \\ {l_1}_X . {k_2}_Y . {l_2}_X . (l_3, l_4)_X \\ {l_1}_X . {k_2}_Y . {l_2}_X . (l_3, l_4)_X . {l_6}_X \end{array}$$

is not complete, because X *cannot respond to* Y *choosing* k_1*:* ${l_1}_X \in S$*, and* ${l_1}_X . {k_1}_Y \in V$ *and* $enabled_X({l_1}_X . {k_1}_Y) \neq \emptyset$*, but there is no move* m *such that* ${l_1}_X . {k_1}_Y . m \in S$*. The strategy would be complete if, for example, the valid position* ${l_1}_X . {k_1}_Y . {l_2}_X . (l_3, l_4)_X . {l_5}_X$ *and all appropriate prefixes were added to the strategy.*

Executions

Executions

- A pair of complete, deterministic strategies

Executions

- A pair of complete, deterministic strategies
- one for each player

Executions

- A pair of complete, deterministic strategies
- one for each player
- defines an execution of the process.

Epistemic restrictions

Epistemic restrictions

- We define two equivalences on valid positions, one for each player.

Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players "know" in the usual (Kripke) way.

Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players “know” in the usual (Kripke) way.
- If s_1 and s_2 are equivalent for Z then $s_1.m$ is in Z’s strategy if and only if $s_2.m$ is in the strategy.

Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players "know" in the usual (Kripke) way.
- If s_1 and s_2 are equivalent for Z then s_1.m is in Z's strategy if and only if s_2.m is in the strategy.
- We are saying that strategies can only be based on what players know.

Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players "know" in the usual (Kripke) way.
- If s_1 and s_2 are equivalent for Z then $s_1.m$ is in Z's strategy if and only if $s_2.m$ is in the strategy.
- We are saying that strategies can only be based on what players know.
- One can design different equivalences to "engineer" the appropriate epistemic concept.

Introspection

Introspection

- An example epistemic restriction: introspection.

Introspection

- An example epistemic restriction: introspection.
- The player knows his own history and what moves were available to him at every point in the past.

Definition 3.16. *For player Z, positions s_1 and s_2 are called Z* indistinguishable *if they satisfy the following conditions:*

(1) $Z(s_1) = Z(s_2)$

(2) $enabled_Z(s_1) = enabled_Z(s_2)$.

(3) *For all prefixes s'_1 of s_1 and s'_2 of s_2, if Z has a move available at both s'_1 and s'_2 and $Z(s'_1) = Z(s'_2)$, then $enabled(s'_1) = enabled(s'_2)$.*

Definition 3.16. *For player Z, positions s_1 and s_2 are called Z* indistinguishable *if they satisfy the following conditions:*

(1) $Z(s_1) = Z(s_2)$

(2) $enabled_Z(s_1) = enabled_Z(s_2)$.

(3) *For all prefixes s_1' of s_1 and s_2' of s_2, if Z has a move available at both s_1' and s_2' and $Z(s_1') = Z(s_2')$, then $enabled(s_1') = enabled(s_2')$.*

Definition 3.17. *Given a process P, and S a strategy for player Z on P, S is* introspective *if for every Z indistinguishable pair of valid positions s_1 and s_2, $s_1.m \in S$ if and only if $s_2.m \in S$.*

$$P = (\nu b)\left(l_1 : \{k_1 : \tau \,.\, l_2 : a \,.\, l_3 : b + k_2 : \tau \,.\, l_2 : c \,.\, l_3 : b\} \mid l_4 : \bar{b} \,.\, (l_5 : d + l_6 : e)\right)$$

the deterministic strategy given above for X*,*

$$\begin{aligned} S \;=\; \{&\varepsilon, \\ & {l_1}_X, \\ & {l_1}_X.{k_1}_Y.{l_2}_Y, \\ & {l_1}_X.{k_2}_Y.{l_2}_X, \\ & {l_1}_X.{k_1}_Y.{l_2}_X.(l_3, l_4)_X, \\ & {l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X, \\ & {l_1}_X.{k_1}_Y.{l_2}_X.(l_3, l_4)_X.{l_5}_X, \\ & {l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X.{l_6}_X\} \end{aligned}$$

is not introspective. This is because in order to satisfy the introspection condition, ${l_1}_X.{k_1}_Y.{l_2}_X.(l_3, l_4)_X$ *and* ${l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X$ *should have the same moves appended to them in* S*, since they are* X *indistinguishable. However,* ${l_1}_X.{k_1}_Y.{l_2}_X.(l_3, l_4)_X.{l_5}_X \in S$ *and* ${l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X.{l_5}_X \notin S$*, and similarly,* ${l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X.{l_6}_X \in S$ *and* ${l_1}_X.{k_2}_Y.{l_2}_X.(l_3, l_4)_X.{l_5}_X \notin S$*.*

$$P = {}^{0}\{\,{}^{1}\tau\,.(\,{}^{3}c\,.(\,{}^{6}f + {}^{7}g\,) + {}^{4}d\,) + {}^{2}\tau\,.(\,{}^{3}c\,.(\,{}^{6}f + {}^{7}g\,) + {}^{5}e\,)\}.$$

Let X's strategy be

$$\begin{aligned} S = \{&\varepsilon, \\ &l_{0X}, \\ &l_{0X}.l_{1Y}.l_{3X}, \\ &l_{0X}.l_{2Y}.l_{3X}, \\ &l_{0X}.l_{1Y}.l_{3X}.l_{6X}, \\ &l_{0X}.l_{2Y}.l_{3X}.l_{7X}. \end{aligned}$$

This strategy is introspective. Even though $X(l_{0X}.l_{1Y}.l_{3X}) = X(l_{0X}.l_{2Y}.l_{3X})$ *and* $enabled_X(l_{0X}.l_{1Y}.l_{3X}) = enabled_X(l_{0X}.l_{2Y}.l_{3X})$*, it is acceptable that the two strings have different moves appended to them, because* $enabled_X(l_{0X}.l_{1Y}) = \{l_{3X}, l_{4X}\}$ *and* $enabled_X(l_{0X}.l_{2Y}) = \{l_{3X}, l_{5X}\}$*. This can be thought of as X being able to distinguish between the two positions* $l_{0X}.l_{1Y}.l_{3X}$ *and* $l_{0X}.l_{2Y}.l_{3X}$ *because he remembers what moves were available to him earlier and is able to use this information to tell apart the two positions.*

Syntactic schedulers

$$
\begin{aligned}
P, Q &::= l : \alpha.P \mid P|Q \mid P + Q \mid (\nu a)P \mid l : \{P\} \mid 0 \\
L &::= l \mid (l, k) \\
\rho, \eta &::= \sigma(L).\rho \mid \mathbf{if}\ L\ \mathbf{then}\ \rho\ \mathbf{else}\ \eta \mid 0 \\
CP &::= P \parallel \rho, \eta
\end{aligned}
$$

$$\text{ACT}\ \frac{}{l:\alpha.P \parallel \sigma(l).\rho,\eta \xrightarrow[l_X]{\alpha} P \parallel \rho,\eta}$$

$$\text{RES}\ \frac{P \parallel \rho,\eta \xrightarrow[s]{\alpha} P' \parallel \rho',\eta' \quad \alpha \neq a,\bar{a}}{(\nu a)P \parallel \rho,\eta \xrightarrow[s]{\alpha} (\nu a)P' \parallel \rho',\eta'}$$

$$\text{SUM1}\ \frac{P \parallel \rho,\eta \xrightarrow[s]{\alpha} P' \parallel \rho',\eta' \qquad \rho \neq \textbf{if } L \textbf{ then } \rho_1 \textbf{ else } \rho_2}{P+Q \parallel \rho,\eta \xrightarrow[s]{\alpha} P' \parallel \rho',\eta'}$$

$$\text{PAR1}\ \frac{P \parallel \rho,\eta \xrightarrow[s]{\alpha} P' \parallel \rho',\eta' \qquad \rho \neq \textbf{if } L \textbf{ then } \rho_1 \textbf{ else } \rho_2}{P|Q \parallel \rho,\eta \xrightarrow[s]{\alpha} P'|Q \parallel \rho',\eta'}$$

$$\text{SWITCH}\ \frac{P \parallel \eta,0 \xrightarrow[j_X]{\tau} P'||\eta',0}{l:\{P\} \parallel \sigma(l).\rho,\eta \xrightarrow[l_X.j_Y]{\tau} P' \parallel \rho,\eta'}$$

$$\text{COM}\ \frac{P \parallel \sigma(l).0,0 \xrightarrow[l_X]{a} P' \parallel 0,0 \qquad Q \parallel \sigma(j).0,0 \xrightarrow[j_X]{\bar{a}} Q' \parallel 0,0}{P|Q \parallel \sigma(l,j).\rho,\eta \xrightarrow[(l,j)_X]{\tau} P'|Q' \parallel \rho,\eta}$$

$$\text{IF1}\ \frac{P \parallel \rho_1,\eta \xrightarrow[s]{\alpha} P' \parallel \rho_1',\eta' \quad P \parallel \sigma(L).0,\theta \xrightarrow[s']{\beta} P'' \parallel 0,\theta' \quad \text{for some scheduler } \theta}{P \parallel \textbf{if } L \textbf{ then } \rho_1 \textbf{ else } \rho_2,\eta \xrightarrow[s]{\alpha} P' \parallel \rho_1',\eta'}$$

$$\text{IF2}\ \frac{P \parallel \rho_2,\eta \xrightarrow[s]{\alpha} P' \parallel \rho_2',\eta' \quad P \parallel \sigma(L).0,\theta \not\rightarrow \quad \text{for all schedulers } \theta}{P \parallel \textbf{if } L \textbf{ then } \rho_1 \textbf{ else } \rho_2,\eta \xrightarrow[s]{\alpha} P' \parallel \rho_2',\eta'}$$

The main technical result

The main technical result

- The introspective restriction exactly captures the independence requirement that one expects

The main technical result

- The introspective restriction exactly captures the independence requirement that one expects
- In particular,

The main technical result

- The introspective restriction exactly captures the independence requirement that one expects
- In particular,
- they are equivalent to the syntactic schedulers of Chatzikokolakis and Palamidessi.

THEOREM 4.4. *Given a deterministically labelled process P, a nonblocking primary scheduler ρ for P, and a nonblocking secondary scheduler η for P, there is a deterministic, complete, introspective X strategy S depending only on P and ρ, and a deterministic, complete, introspective Y strategy T depending only on P and η, such that the execution of $P \parallel \rho, \eta$ is identical to the execution of P with S and T.*

Furthermore, given a deterministically labelled process P, a deterministic, complete, introspective X strategy S for P, and a deterministic, complete, introspective Y strategy T for P, there is a nonblocking primary scheduler ρ depending only on S and P and a nonblocking secondary scheduler η depending only on T and P such that the execution of P with S and T is identical to the execution of $P \parallel \rho, \eta$.

Probabilistic Choice

Probabilistic Choice

- Chatzikokolakis and Palamidessi also defined schedulers for a probabilistic process algebra.

Probabilistic Choice

- Chatzikokolakis and Palamidessi also defined schedulers for a probabilistic process algebra.
- We have formalized this also and proved a similar correspondence theorem.

Modal Logic

Modal Logic

- Epistemic concepts are nicely captured by an S5 modal logic.

Modal Logic

- Epistemic concepts are nicely captured by an S5 modal logic.
- Such logics have been very useful in the theory of distributed systems but have been slow to penetrate concurrency theory.

Modal Logic

- Epistemic concepts are nicely captured by an S5 modal logic.
- Such logics have been very useful in the theory of distributed systems but have been slow to penetrate concurrency theory.
- We present a modal logic for capturing the notion of introspection and other epistemic aspects of the agents.

Knowledge

Knowledge

- Usually modelled with an equivalence relation on the set of states (possible worlds), which represents what the agents thinks is possible.

Knowledge

- Usually modelled with an equivalence relation on the set of states (possible worlds), which represents what the agents thinks is possible.
- If St is the set of states then the agent knows phi in state s if for all states t with s~t, phi is true in t.

Axioms for Knowledge

1. All propositional tautologies
2. $(K_i\phi) \wedge (K_i(\phi \Rightarrow \psi)) \Rightarrow K_i\psi$
3. $K_i\phi \Rightarrow \phi$
4. $K_i\phi \Rightarrow K_iK_i\phi$
5. $\neg K_i\phi \Rightarrow K_i(\neg K_i\phi)$
6. *Modus Ponens*
7. From ϕ infer $K_i\phi$

Some Remarks

Some Remarks

- There are variant axiomatizations possible.

Some Remarks

- There are variant axiomatizations possible.
- The axioms given correspond to assuming that the possibility relation is an equivalence relation.

Some Remarks

- There are variant axiomatizations possible.
- The axioms given correspond to assuming that the possibility relation is an equivalence relation.
- The axioms given are for a static situation.

Some Remarks

- There are variant axiomatizations possible.
- The axioms given correspond to assuming that the possibility relation is an equivalence relation.
- The axioms given are for a static situation.
- Many combinations are possible: time, probability, dynamic update.

What our logic should say

—Which player made the last move and what the last move was,

—What moves are available and what player they belong to,

—What formulas are satisfied by specific continuations of the current valid position,

—What formulas are satisfied by specific prefixes of the current valid position,

—The knowledge of each player in the current state, according to the introspective indistinguishability condition discussed in section 3, and

—What formulas were satisfied by the state immediately after either player's last move.

Let L represent a general label (a single label or a synchronizing pair of labels), m a move (a general label together with a player), let X and Y be the two players, and let Z represent either X or Y.

$$\phi ::= C_Z(L) \mid A_Z(L) \mid \bigcirc_m \phi \mid \circleddash\phi \mid K_Z\phi \mid @_Z\phi \mid \phi \wedge \phi \mid \neg\phi \mid \top.$$

Let L represent a general label (a single label or a synchronizing pair of labels), m a move (a general label together with a player), let X and Y be the two players, and let Z represent either X or Y.

$$\phi ::= C_Z(L) \mid A_Z(L) \mid \bigcirc_m \phi \mid \circleddash\phi \mid K_Z\phi \mid @_Z\phi \mid \phi \wedge \phi \mid \neg\phi \mid \top.$$

$C_Z(L)$: last move was L_Z.

Let L represent a general label (a single label or a synchronizing pair of labels), m a move (a general label together with a player), let X and Y be the two players, and let Z represent either X or Y.

$$\phi ::= C_Z(L) \mid A_Z(L) \mid \bigcirc_m \phi \mid \circleddash\phi \mid K_Z\phi \mid @_Z\phi \mid \phi \wedge \phi \mid \neg\phi \mid \top.$$

$C_Z(L)$: last move was L_Z.

$A_Z(L)$: L_Z is available now.

Let L represent a general label (a single label or a synchronizing pair of labels), m a move (a general label together with a player), let X and Y be the two players, and let Z represent either X or Y.

$$\phi ::= C_Z(L) \mid A_Z(L) \mid \bigcirc_m \phi \mid \circleddash\phi \mid K_Z\phi \mid @_Z\phi \mid \phi \wedge \phi \mid \neg\phi \mid \top.$$

$C_Z(L)$: last move was L_Z.

$A_Z(L)$: L_Z is available now.

$\bigcirc_m\phi$: after move m, ϕ will be true;
it asserts that m is available.

Let L represent a general label (a single label or a synchronizing pair of labels), m a move (a general label together with a player), let X and Y be the two players, and let Z represent either X or Y.

$$\phi ::= C_Z(L) \mid A_Z(L) \mid \bigcirc_m \phi \mid \ominus\phi \mid K_Z\phi \mid @_Z\phi \mid \phi \wedge \phi \mid \neg\phi \mid \top.$$

$C_Z(L)$: last move was L_Z.

$A_Z(L)$: L_Z is available now.

$\bigcirc_m\phi$: after move m, ϕ will be true;
it asserts that m is available.

$\ominus\phi$ means that ϕ was true at the previous valid position

Let L represent a general label (a single label or a synchronizing pair of labels), m a move (a general label together with a player), let X and Y be the two players, and let Z represent either X or Y.

$$\phi ::= C_Z(L) \mid A_Z(L) \mid \bigcirc_m \phi \mid \ominus\phi \mid K_Z\phi \mid @_Z\phi \mid \phi \wedge \phi \mid \neg\phi \mid \top.$$

$C_Z(L)$: last move was L_Z.

$A_Z(L)$: L_Z is available now.

$\bigcirc_m\phi$: after move m, ϕ will be true;
it asserts that m is available.

$\ominus\phi$ means that ϕ was true at the previous valid position

$K_Z\phi$ means Z knows ϕ.

Let L represent a general label (a single label or a synchronizing pair of labels), m a move (a general label together with a player), let X and Y be the two players, and let Z represent either X or Y.

$$\phi ::= C_Z(L) \mid A_Z(L) \mid \bigcirc_m \phi \mid \circleddash\phi \mid K_Z\phi \mid @_Z\phi \mid \phi \wedge \phi \mid \neg\phi \mid \top.$$

$C_Z(L)$: last move was L_Z.

$A_Z(L)$: L_Z is available now.

$\bigcirc_m\phi$: after move m, ϕ will be true;
it asserts that m is available.

$\circleddash\phi$ means that ϕ was true at the previous valid position

$K_Z\phi$ means Z knows ϕ.

$@_Z\phi$ means ϕ was true just after Z's last move.

THEOREM 6.2. *$s \sim_Z t$ if and only if s and t agree on all formulas of the form*

$$(@_Z \circleddash)^n @_Z C_Z(L)$$

for $n \geq 0$, and for any L, and also agree on all formulas of the form

$$(@_Z \circleddash)^n A_Z(L)$$

for $n \geq 0$ and for any L.

Conclusions

- We have shown that the syntactic restrictions of Chatzikokolakis and Palamidessi can be viewed as semantic restrictions on the strategies allowed.
- It is easy to impose other restrictions if one wants; it is not so easy to define a new syntax and operational semantics for schedulers every time one wants to consider a variation.
- Epistemic concepts are pervasive in security; they should be made manifest.

Dreams

- Epistemic logic and information theory should fuse to give a new quantitative theory of information flow.
- Process algebra should be enriched to allow more subtle interactions (e.g. games) between agents.